Design

By	BD	
Regarding	Editform Ajaxers	
To	Team	
Date	15JAN15	
Status	In commercial confidence; distribution to relevant project team members and selected 3rd parties allowed	
History	V1.0, 15JAN15	BD

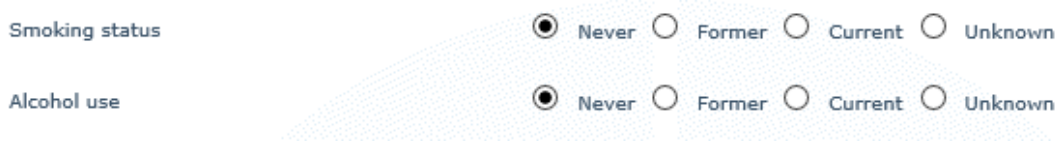
Synopsis

This document describes a feature that has recently been added to edit forms: Ajaxers. Note that at the time of writing this feature has not (yet) been implemented for grid- and matrix forms.

Ajaxers allow an edit form to change dynamically based on the input by the user.

The following is an example that demonstrates how Ajaxers can be used to make forms more intuitive.

Imagine a system where we prompt for the smoking status of a patient. 'Never' is a perfectly valid answer that requires no further explanation.



However, as the user clicks 'Former' or 'Current' we need to prompt for additional information related to the smoking status. A new question 'magically' appears asking the user for the 'Number of pack years' of the patient.

This extra question will disappear again as soon as the user selects 'Never' again.

Ajaxer Actions

There are a number of things you can do with Ajaxers (referred to as 'actions'):

- When a specific condition is met (as far as user input is concerned)
  - Hide / show a row (a row includes the label, actual field, any postlabel and refs; a row can also contain merged fields)
  - Hide / show a sub-title for a field
  - Hide / show a label associated with a field
  - Hide / show a field itself
  - Change the class / style for a row, sub-title, label or field
  - Change the label or sub-title
  - Set a field to a specific value
  - Execute some bespoke JavaScript

Technical Implementation

Ajax functionality can be implemented by assigning handlers to certain events on fields. This can however be somewhat fiddly; different browsers handle events in a different way and multiple events may have to be set-up if an action depends on the input of multiple fields.

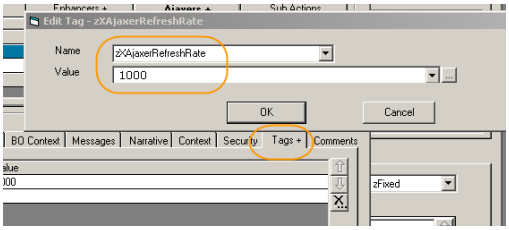
Instead, CaseMaster® implements Ajaxers in a way that can be discarded as somewhat unconventional.

We basically execute a small piece of logic every x milliseconds (standard 500ms but can be overruled). In this piece of logic we check for certain conditions and if these are met, we apply the necessary actions. This feels brute force as it will

repeat the same logic every 500ms. Our experience is however that local processing power is such that is not a problem for todays' computers / browsers.

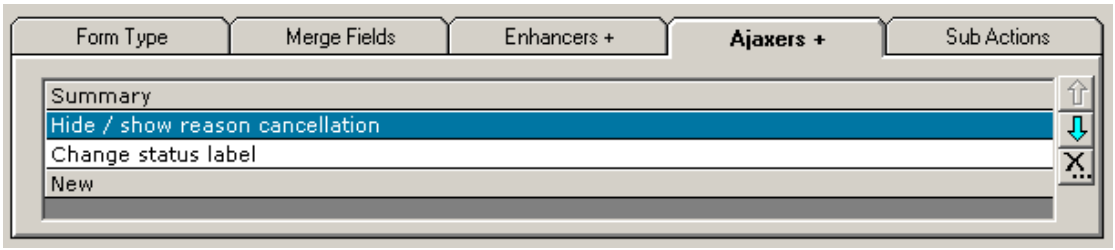
Changing the Refresh Rate

You may want to alter the refresh rate (the rate at which we check for conditions to be met) from the default 500ms. You do this by setting the tag 'zXAjaxerRefreshRate' to the number of milliseconds you want.



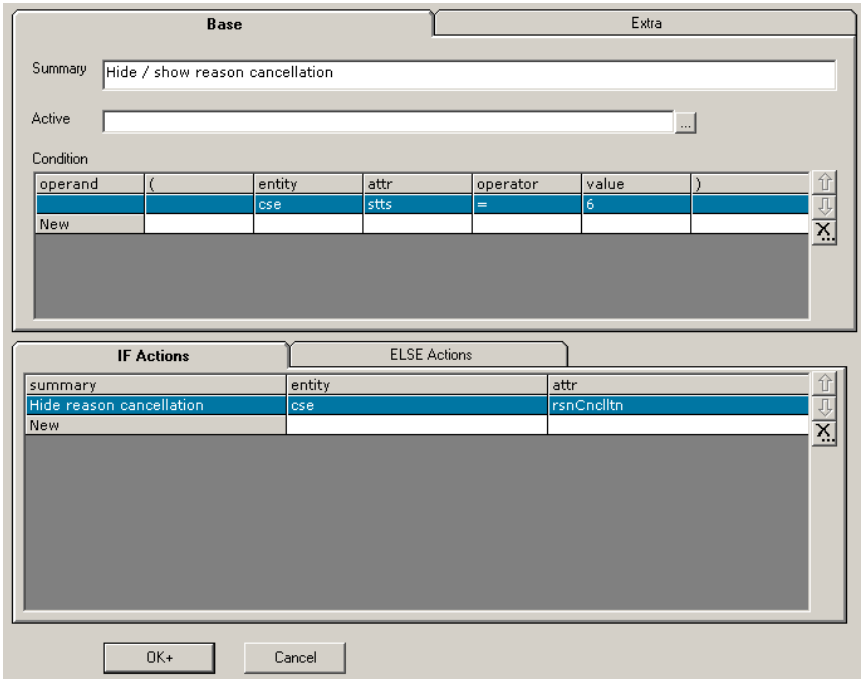
Defining Ajaxers

Each edit form can have any number of Ajaxers. You can manage them from the 'Ajaxers' tab on an edit form.



One Ajaxer consists of a condition and any number of actions to take when the condition is met and actions when the condition is not met.

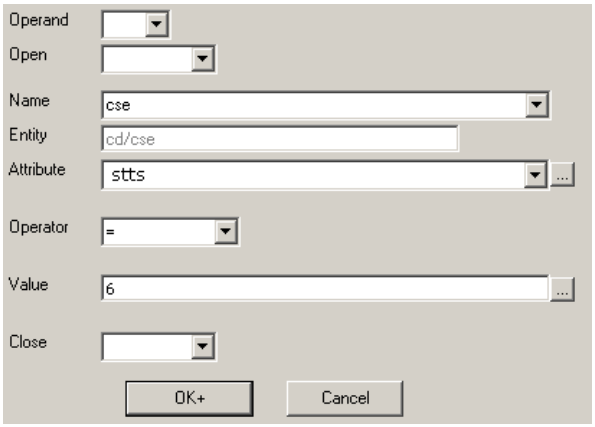
Each Ajaxer must have a summary. This is some plain text (can thus contain funny characters, it is not used to generate code) that explains what the Ajaxer does. Optionally you can be more verbose by entering comments (under the 'Extra' tab.



The optional active expression can be used to specify when the Ajaxer is relevant or not. Note that this active expression is evaluated on the server and simply means: shall this Ajaxer be included in the HTML or not.

The condition is the actual logic that is included in the HTML and that will, in near real-time, check for the input on the screen.

A simple condition only has one line; comparing the value of a field with certain input. More complex conditions can have multiple lines using 'AND' and 'OR' and parenthesis to construct a complex condition.

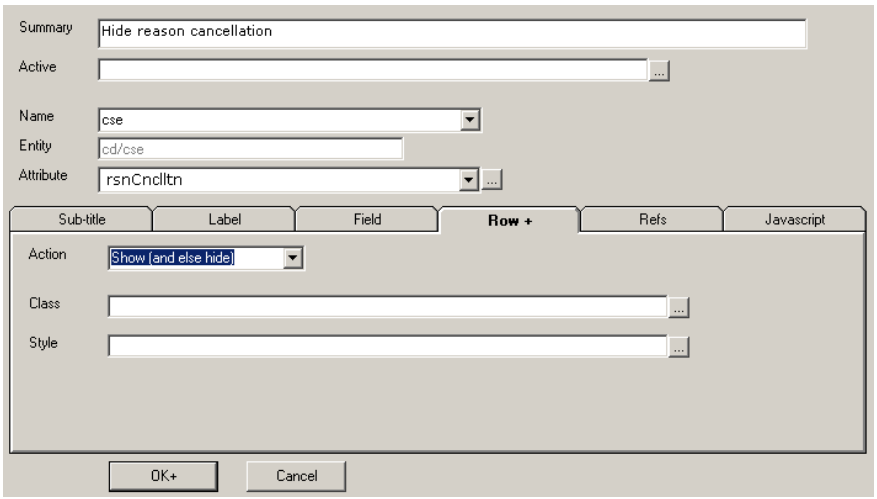


In our simple condition we check whether the field cse.stts has the value 6 (in our example 6 is the value for 'Cancelled').

Note that the value can be entered as a director and that values such as #date and #user can be used here. There is no need to take the datatype of the field into consideration; this is done automatically by CaseMaster®.

The next step is define what actions need to be executed when this condition is met (IF actions) and / or what actions need to be taken when the condition is not met (ELSE actions).

Our action is simple: when the user selects the value 'Cancelled' from the list of statuses (cse.stts = 6), we need to show the field 'Reason cancellation' and otherwise we would need to hide this field.



We could create two very similar actions; one as IF-action and one as ELSE-action. The IF-action would show the row of the reason cancellation field (the row includes the label, field, any postlabels and refs) and the ELSE-action would hide it.

This however is a very common pattern (show IF and hide ELSE) that it would be a shame to have to create two actions. Instead we can use the action 'Show (and else hide)' a shortcut for creating a separate show and hide action.

You can again use an active expression to exclude an action from being included in the HTML.

Important Notes

Ajaxers and Validations

Showing an hiding a field does not mean that that system will validate for correct input. In our example, the user can select 'Cancelled' without entering a reason for cancellation. Showing the field does not make it mandatory. You will still have to create a validation rule to check for valid input.

Refs

At the time of this writing Ajaxer actions cannot be applied to refs.

<End of document>